kx-trees: An unsupervised learning method for use in developmental agents

Brandon Rohrer
Intelligent Systems, Robotics, and Cybernetics Group
Sandia National Laboratories, Albuquerque, NM 87185-1010
Email: brrohre@sandia.gov
Web: www.sandia.gov/∼brrohre

***Abstract*—Acquiring concepts from experience is a key aspect of development and one that is commonly neglected in learning agents. In this work, concept acquisition is formulated as an unsupervised learning problem and is addressed with a novel algorithm: kx-trees. kx-trees differ from prior approaches to unsupervised learning in that they require very little information; four user selected parameters determine all aspects of kx-trees' performance. Notably, and in contrast with most other unsupervised learning approaches, they do not require that the input state space be well-scaled. kx-trees' operation is described in detail and illustrated with two simulations. The second simulation shows some similarities between kx-trees and feature construction in the human visual processing system.**

I. Introduction

In developmental agents, creating abstract representations is an important aspect of learning. Unless an agent starts out with a pre-programmed set of concepts, it must generate them from experience. One method for doing this is to group observations into clusters, which clusters are then treated as concepts. Clustering state observations is a type of unsupervised learning. (A wide-ranging review of unsupervised learning methods can be found in [1, ch. 10].) Unsupervised learning allows agents to create higher-level concepts from their experiences. Expressing raw experiences in more abstract terms greatly decreases the size of the agent's input space. This in turn lets the agent use more powerful learning, perception, and planning algorithms that can only operate effectively in modest state spaces.

In this work, state observations are grouped into a small number of clusters. Each dimension in the state space represents a separate state variable. What separates this work from previous unsupervised learning methods is that it does not assume that state dimensions are ordered fields. That is, it does not assume that 1 is less than 2 or that the distance from 1 to 2 is less than the distance from 1 to 1000. As a result, this method can be applied to a wide variety of data, including continuous, discretized, categorical, binary, and hybrid data types.

The algorithm used here for clustering possibly non-ordered fields is a binary decision tree. Initially the entire state space represents a single cluster. The state space is then repeatedly subdivided into regions, with some of the regions representing clusters. Each subdivision is represented by splitting a leaf node into two new leaves. A subset of the tree's leaves are the clusters. The dimension along which to divide each leaf is chosen so as to maintain and isolate clusters as much as possible. A tree that chooses when and where to divide solely on the basis of the data distribution is said to have the X-property [2]. In order to distinguish between this and other tree-based tools (including Decision Q-Trees [3], S-trees [4], and T-Trees, an extension of Continuous U-Trees [5]) it will be referred to as an kx-tree.

In this paper, I present a detailed description of kx-trees, together with two implementations demonstrating their operation. One kx-tree is used to find clusters in a two-dimensional state space. A second kx-tree is used with digital images to find visual primitives similar to those found in human cortical area V1.

II. Method

kx-trees are conceptually similar to other decision tree based algorithms, but differ in the specifics of their implementation. Although the most well known implementations of decision trees are CART [6] and C4.5 [7], a concise overview of a wide variety of decision trees is given in [8]. A decision tree is defined by how it answers the three questions 1) When to split a node, 2) Where to split that node, and 3) When to stop splitting a node. (Some trees also answer a fourth question—When and how to prune branches from the tree—but kx-trees do not use pruning.) After describing the structure and populating process of a leaf, this section answers those questions for kx-trees.

A. *Leaf structure*

Each leaf represents an explicitly bounded region of the state space. When defining the initial leaf that encompasses the entire state space, upper and lower bounds on all input dimensions must be known or assumed. As a leaf is divided, the bounds of its children are created using the decision boundary. In this way, the leaves form a non-overlapping set of regions that completely cover the state space. (See Figure 1) Each leaf is a hyperbox, defined by minimum and maximum values in each dimension of the state space.

Each leaf represents a single region. Each node on the tree represents a super-region, the combination of the regions defined by its children, which themselves may be either leaves or branch nodes. Most tree-based learning tools are supervised

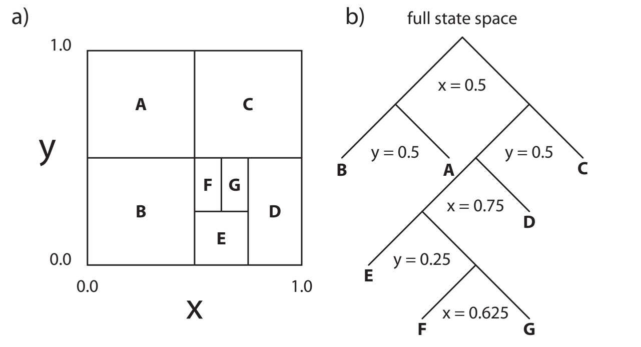


Fig. 1. **a)** A partitioned state space and **b)** the decision tree representation of that partitioning. The top node in the tree represents the entire state space. It also indicates the division of the space into its left and right halves along the line $x = 0.5$. Leaf nodes in the tree represent individual regions as labeled. Any internal node on the tree represents the union of all of the leaf node regions that fall under it.

learning algorithms, rather than unsupervised. Observations of input states are typically accompanied by categories, in classification problems, or values, in regression problems. kx-trees differ from these. The leaves of a kx-tree do not provide an estimate of a reward, value function, or class membership. Instead, kx-trees identify locally dense regions of state activity. In addition, kx-trees subdivide clusters once they exceed a predetermined size. The motivating assumption behind kx-trees' partitioning heuristic is that useful concepts capture a roughly fixed number of observations. If a concept represents very few observations, it provides only limited abstraction, and if it represents a very large number of observations, it may be too general to be useful.

B. When to split a node

A kx-tree node becomes eligible to split when the number of observed states it contains exceeds a user-defined threshold, M. Each observation that falls within a given region of state space is a *member* of that region. When a node has more than M members, it is considered ripe for bisection. However, each member has a finite lifetime, L, also defined by the user. As a result, a node is ready to split when more than M of the last L state observations fall within its region of state space.

C. Where to split a node

The objective of a kx-tree is to find clusters of observations and bound them tightly. Each cluster represents a concept, a feature, or a closely related group of states. Tighter bounds give a more concise concept definition. Therefore, when a node is bisected, the dimension along which to split and the value at which to split it are chosen such that the cluster is preserved as much as possible. For all admissible splits, the split which maximizes the disparity between the number of members in each child is selected. If more than one candidate split achieves the maximum value, then the split value is randomly selected from among them.

More formally, for an admissible split (d, x) along dimension d at value x, the quality of the split, q is given by

$$q_{d,x} = |m_a - m_b| \tag{1}$$

where m_a is the number of members from the parent node that would fall under one of its children for that split and m_b is the number of members that would fall under the other child node. Quality is symmetric with respect to m_a and m_b. The winning split fulfills the condition

$$(d, x)_{\text{winner}} = \max_{d \in D}(\max_{x \in X_d} q_{d,x}) \tag{2}$$

where D is the set of all state space dimensions and X_d is the set of all admissible split values for dimension d. The admissible split values in X_d divide the region along dimension d roughly in half. X_d is generated by taking a number of evenly-spaced values along d. More specifically, if $d_{\min}$ and $d_{\max}$ are the limits of d in the region to be divided, then N admissible split values can be generated:

$$\begin{aligned} d_{\text{low}} &= d_{\min} + \frac{1}{4}(d_{\max} - d_{\min}) && (3) \\ d_{\text{high}} &= d_{\min} + \frac{3}{4}(d_{\max} - d_{\min}) && (4) \\ x_i &= d_{\text{low}} + (d_{\text{high}} - d_{\text{low}})\frac{i}{N+1}, \forall i \leq N && (5) \end{aligned}$$

The resulting values of X_d fall between, but do not include, one quarter and three quarters of the range of d. N is a user-selected constant.

D. When to stop splitting a node

Due to the fact that members have a limited lifetime, an explicit stopping criterion is typically unnecessary. The splitting condition of accumulating more than M members within any interval L becomes harder to achieve as regions get smaller. With any continuous distribution of observations, the member accumulation rate will approach zero as region size approaches zero. However, when processing discontinuous distributions (as with categorical or discretized data), observations may be grouped at a single value, prompting endless repeated splits. This can occur whenever a region's range in a given dimension is less than the discrete resolution in that dimension. A method for handling this degenerate case is to specify a threshold for the minimum region size in each dimension. This can either be a constant ϵ that is the same for each dimension or a vector E with separate values for each dimension. Further divisions are not allowed along any dimension for which $d_{\max} - d_{\min} < \epsilon$. The same effect could be achieved by adding a small amount of noise (ϵ) to each observation.

E. Clusters

kx-trees seek to isolate groups of members into clusters. A leaf node becomes a cluster when it collects more than $M/2$ members. Thus, when a node is divided, one of the children will become a cluster and the other will not (although it may eventually become one too), and the parent node ceases to be a cluster. Once a leaf node is designated a cluster, it remains one, even if the number of members it contains falls below $M/2$. Cluster nodes represent concepts, the categories into which most of the data fall, or features which are common in the data.

Once clusters have been formed, they serve as a means of interpreting new states. A single state observation is a point in state space; if it falls within the hyperbox of a cluster, the data point can be represented more abstractly as that concept. But typically, clusters do not provide complete coverage of the state space. In order to find a conceptual interpretation of points that do not fall within a cluster, a similarity or distance measure can be used to identify the nearest clusters. The similarity measure kx-trees use reflects the fraction of dimensions in which an observation matches a cluster. The similarity, σ, between a state s and a cluster c in an n-dimensional space, $\mathcal{S}$, is given by

$$\sigma(s, c) = \frac{\sum_{i=1}^{n} \text{ in}(s, c, i)}{n} \tag{6}$$

where

$$\begin{aligned} \text{in}(s, c, i) &= 1, \text{ if } s_i \geq c_{i\,\min} \text{ and} \\ & \quad\; s_i < c_{i\,\max} \\ &= 0, \text{ otherwise} \end{aligned} \tag{7}$$

where s_i is the value of s, $c_{i\,\min}$ is the minimum value of c, and $c_{i\,\max}$ is the maximum value of c, all in the i^{th} dimension. The corresponding distance measure, δ, is given by

$$\delta(s, c) = 1 - \sigma(s, c) \tag{8}$$

If s falls within c for all dimensions, then $\delta = 0$ and $\sigma = 1$. If s falls within c in only half of the dimensions, $\delta = \sigma = 1/2$. The distance measure takes no account of how far outside a cluster a state may fall. Only matching and non-matching are reflected in the distance. In this way, the distance measure is similar to Hamming distance, but is extended to apply to real valued vectors and is normalized to fall within the range $[0, 1]$. In fact, for binary state spaces, the modified Hamming distance of Equation 8 reduces to the actual Hamming distance divided by the number of state space dimensions, n. It should be noted that this is a departure from many other unsupervised learning approaches, which use Euclidean distance. The modified Hamming distance and modified Hamming similarity avoid making the assumption that the state space is well scaled or that its dimensions are ordered fields, and this allows it to be applied to a wider range of problems.

F. Coordinate transformation to feature space

Combined with the modified Hamming similarity, clusters can be used to reinterpret states. The vector of similarities between a state and each of the clusters translates that state into a new state space, $\mathcal{S}'$, in which each dimension represents a cluster. The new state, s', is given by

$$s' = [\sigma(s, c_1), \sigma(s, c_2), \ldots \sigma(s, c_{n'})] \tag{9}$$

where n' is the total number of clusters and the dimensionality of the new state space.

Multiple clusters may lie very near each other in $\mathcal{S}$, such that a state falling completely within one cluster may have high similarity to other clusters as well. This would cause dimensions in $\mathcal{S}'$ to behave as if they were highly non-orthogonal.

A pseudo-orthogonal state space can be formed by using a winner-take-all projection onto $\mathcal{S}'$:

$$s' = [0, 0, 0, \ldots, \sigma(s, c_i), \ldots, 0, 0, 0] \tag{10}$$

where similarity for the i^{th} dimension is a maximum:

$$\sigma(s, c_i) \geq \sigma(s, c_j), \forall j \neq i \tag{11}$$

If this condition holds for more than one dimension, the winner is randomly selected from among them. By forcing s to project onto only one dimension in $\mathcal{S}'$, the relative contributions of neighboring clusters remain clearly distinguishable.

Transforming observations from $\mathcal{S}$ to $\mathcal{S}'$ re-expresses them in terms of more complex features of the input space. This process converts low-level data into somewhat higher-level features. In the cases where $n' < n$, it also results in a dimensionality reduction.

It is noteworthy that the inputs and the outputs of kx-trees' coordinate transformation are both in the same form: vectors with real-valued elements. As a result, multiple kx-trees can be arranged hierarchically with the output of one serving as input to another. It is expected that with each level in the hierarchy, the feature representation would become increasingly abstract. This phenomenon is the subject of current research.

III. Simulations and Results

A. Simulation 1: Clustering example

The first simulation reported here is a simple but illustrative example. In it, a pair of arbitrarily selected two-dimensional normal distributions were used to generate random points in a two-dimensional state space. The defining parameters for the kx-tree were chosen as follows:

Anomaly lifetime,	L	$=$	100
Maximum number of members,	M	$=$	40
Number of split candidates per dim.,	N	$=$	101
Minimum category size,	ϵ	$=$	0

The kx-tree processed the points as they were generated, creating a partition of the state space and finding the cluster leaves associated with it.

The results for simulation 1 are shown in Figure 2. The two normal distributions produced two groups of points, and the recursive partitioning of the space produced cluster nodes corresponding to the densest portion of each group. The tree in Figure 3 provides an alternative representation of the partition in Figure 2 and shows two major branches, one corresponding to each of the clusters. Figure 4 shows how the state space,

$\mathcal{S}$, maps onto the feature space, $\mathcal{S}'$, defined by the clusters. Simulation 1 was run until a stable tree structure was achieved, that is, no new nodes were created for a substantial period of time. Simulation 1 provides an illustration of kx-tree's operation on a simple problem and is intended to provide an intuitive basis for understanding the results of simulation 2.

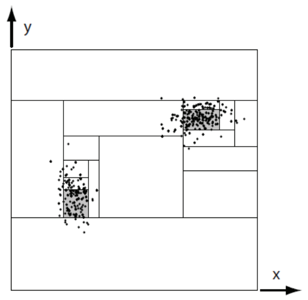


Fig. 2. Two dimensional state space as partitioned by an kx-tree. The two normal distributions produced clearly separated groups of points. Each bisection is shown by a line on the bisection boundary. Each region represents a leaf node. Cluster nodes are shaded gray and correspond to the densest part of each data distribution.

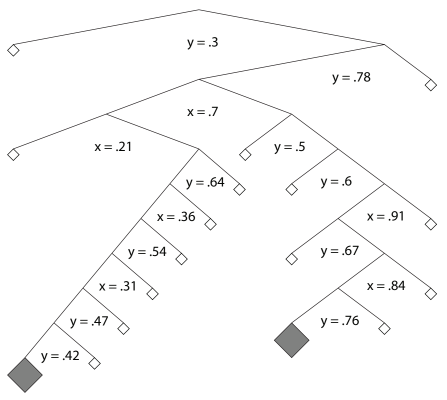


Fig. 3. Tree representation of the partitions shown in Figure 2. The root node of the tree represents the entire state space. Each branching represents a split at the line given by the corresponding equation. The right branch represents the region to the right or above the split. Unfilled boxes on leaf nodes show regions that are not clusters. The two gray filled boxes show the clusters.

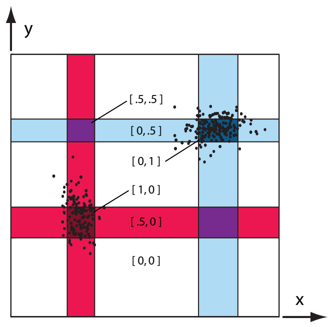


Fig. 4. Mapping from state space $\mathcal{S}$ to feature space $\mathcal{S}'$. Values of s' are given for each region in s, by color.

B. Simulation 2: Vision

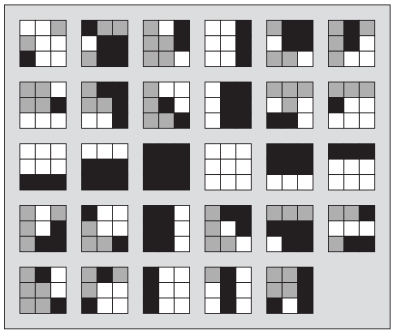
In the second simulation, kx-trees were applied to grayscale images to find simple visual features. The simulation was designed to be similar to the human visual processing system. The images were taken from the CalTech-256 image database [9]. Images were randomly selected, Gaussian blurred with a pixel radius of 6, then downsampled by a factor of 4 (such that the resulting image had 1/16 the number of pixels). This was done to remove the small-scale artifacts of JPEG compression. Then, images were convolved with a center-surround difference filter in order to mimic the retinal and thalamic processing that results in center-surround receptive fields for neurons entering layer 4 of cortical area V1 [10]. Each pixel value in the resulting center-difference image was obtained by taking the original pixel value and subtracting a fraction of the value of each of its neighbors: one-sixth for the four pixels with which it shares a border and one-twelfth for its four diagonal neighbors. The resulting difference value had the range of $[-1, 1]$. Three-by-three pixel groups were taken in a systematic raster tiling from the difference image and processed using an kx-tree. After the image had been completely scanned, a new image was selected and the process repeated. The nine-dimensional state space was partitioned and clusters were identified. The defining parameters for the kx-tree were chosen as follows:

Anomaly lifetime,	L	$=$	1000
Maximum number of members,	M	$=$	70
Number of split candidates per dim.,	N	$=$	1
Minimum category size,	ϵ	$=$	1.0

Choosing $N = 1$ ensured that all splits would be perfect bisections, and choosing $\epsilon = 1.0$ ensured that no dimension would be split more than once. The resulting partition had a maximum of $2^9(512)$ distinct regions.

The result of running simulation 2 for 200,000 observations was a set of clusters in the nine-dimensional state space, shown in the top panel of Figure 5. As shown in the bottom panel, these can be notionally compared to the receptive fields identified in the human visual cortex, which have exhibited strong characteristics of directionality [10]. While the correspondence is too loose to be conclusive, it suggests that kx-trees may produce feature representations that are similar to those produced by the human neocortex.

A difference image can be reconstructed from the feature space representation by breaking it into 9-pixel tiles, transforming each tile from $\mathcal{S}$ to $\mathcal{S}'$, and then transforming them back to $\mathcal{S}$. The forward transformation given in Equation 9 was used here. The inverse transformation from $\mathcal{S}'$ to $\mathcal{S}$ was achieved by finding the centroid of each cluster hyperbox, weighting it by the value of that dimension in s', and linearly summing the results for all clusters. The results of the inverse transformation illustrate which elements are captured in the feature representation. A richer feature set will result in a higher-fidelity reconstructed image. The results of three such reconstructions are shown in Figure 6.



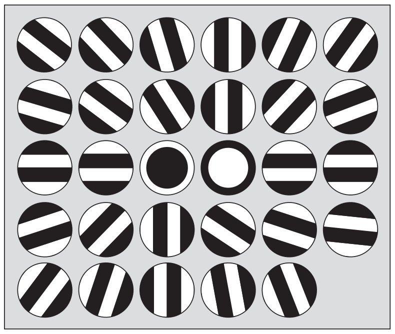


Fig. 5. Top: Graphical representation of the 29 clusters found in the space of 9 pixel groups. In each cluster, dimensions (pixels) that have been bisected are represented as white or black, depending on whether the cluster falls above or below the bisection. Pixels in gray have not been bisected. Bottom: Strongly oriented stimuli, matched to each cluster above. Most of the clusters have unbisected pixels, and hence ambiguous receptive fields. While not unique, these stimuli would each be a match for their corresponding cluster in the top panel.

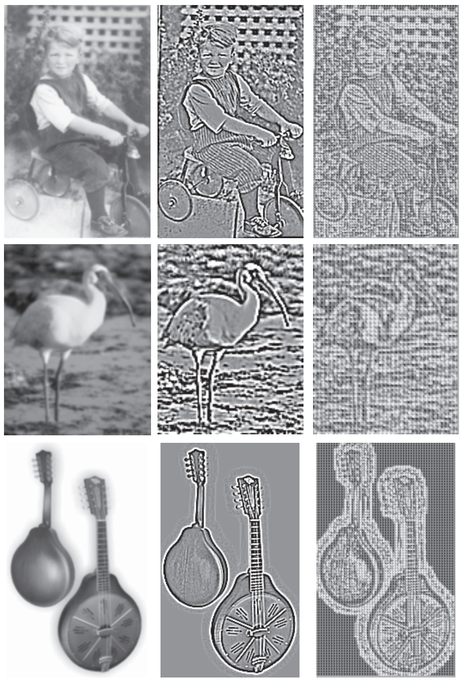


Fig. 6. Original grayscale images (left, taken from [9]), difference images (center), and reconstructed difference images (right), based on the features from Figure 5. The difference images show the derivative of the gradient in the images and mimic the information retained by the center-surround processing performed by the human visual system. The reconstructed images retain the larger-scale features from the difference images.

IV. Discussion

Concept acquisition is an important aspect of development and learning, clearly evident in the cognitive processes of humans and animals. In this work, concepts are defined as clusters of lower-level observations. While this definition does not satisfy all theories of what a concept is, it does provide a useful basis for beginning an investigation of the topic. Formulated this way, the problem of concept acquisition becomes a clustering or unsupervised learning problem. Acquired concepts can be high level and very abstract, or low level and quite specific. In either case, clusters of state observations are identified and used to represent the underlying states.

A. Relation to other work

A decision tree is a natural representation for the iterative division of categories. Each leaf represents a terminal category. Each node represents both the super-category formed by combining its two children, as well as the decision boundary used to differentiate between them. (See Figure 1) Several tree-based adaptive partitioning algorithms have been previously proposed. The Parti-Game algorithm [11] uses a greedy controller to crawl through a partitioned state space. When the controller fails, the last visited subspace is divided. Parti-Game is specifically geared to geometric path planning; it assumes that all paths through the state space are continuous. The G Algorithm [12], U-Tree [13], Continuous U-Tree [14], AMPS [15], and decision trees of Pyeatt and Howe [16] are all approaches used in conjunction with dynamic programming methods or the popular temporal difference method, Q-learning [17], to estimate the value function across the state space. Whenever a subspace's value estimate is shown to be inadequate, the subspace is divided. The G Algorithm handles binary data, U-Trees handle discrete data, and Continuous U-Trees, AMPS, and Pyeatt and Howe's approach handle continuous data.

Taking a broader view, kx-trees are a member of the set of unsupervised learning methods, that is, methods that group data that is unlabeled and unclassified. Also referred to as clustering algorithms, there are many unsupervised learning methods developed with different sets of assumptions, but kx-trees provide a novel collection of characteristics. kx-trees are an on-line, hierarchical, divisive clustering algorithm. They are stable in the sense that cluster boundaries do not move. kx-trees are most notable for what they don't assume. Unlike many clustering algorithms, kx-trees operate without assuming 1) how many categories exist in the underlying data, 2) that prior probabilities of any category are known, 3) that prior probabilities are stationary, and 4) that the forms of the class conditional probabilities are known. Once the dimensionality and range of the state space is defined, kx-trees always start from the same initial conditions, so there is no need to carefully pick initial values for cluster parameters. Only the four operating parameters L, M, N, and ϵ need to be selected. One of kx-trees' greatest strengths is that it does not assume that the input space is well scaled or even linear. In fact, kx-trees' distance metric, the modified Hamming distance, does not even assume that each of the input dimensions is an ordered field. As a result, kx-trees can handle inputs consisting of enumerated types, such as {vanilla = 1, chocolate = 2, strawberry = 3, pistachio = 4}. And their computational demands are modest.

kx-trees' strength comes at a cost, of course. They bound clusters with hyperboxes, so clusters that are not well fit by a hyperbox may not be concisely represented. If M is large, kx-trees can require a relatively large amount of data before useful clusters are generated. They are unstable in the sense that the total number of clusters created can, under certain conditions, grow without bound if not artificially limited. Like other unsupervised learning methods in which the number of clusters is not specified, the validity of the clusters found depends entirely on the appropriateness of the measure of cluster goodness (the node-splitting criterion in the case of kx-trees). And, like other unsupervised learning algorithms of its class, there are no theoretical performance guarantees.

B. Relation to other biologically motivated methods

The performance of kx-trees on identifying visual features similar to the receptive fields of V1 is suggestive, but inconclusive. It indicates that further investigation of kx-trees as a biological learning and concept acquisition mechanism is warranted.

There are several other approaches to the problem of concept acquisition that are motivated by theories of human psychology and neuroscience. Hierarchical Temporal Memory (HTM) [18] uses on a hierarchy of Bayesian classification elements to cluster data into concepts ("causes") and to infer likely causes for a given set of inputs. Adaptive Resonance Theory (ART) [19] uses neural networks to bound clusters of states within hyperboxes. Like kx-trees, both HTMs and ART are online methods, meaning that they can incorporate new state observations and update their results incrementally. However, HTMs and ART differ from kx-trees in that they use a Euclidean distance metric to determine the similarity between states. For Euclidean distance to be valid, the state space must be well scaled, that is, a unit step in each dimension must represent the same magnitude of change. This is a somewhat restrictive condition that kx-trees do not need to meet. Because kx-trees do not use Euclidean distances (or any p-norm), they do not require well-scaled state spaces.

Acknowledgments

This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories. Sandia National Laboratories is a multi program laboratory operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

[1] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. Wiley Interscience, 2001.

[2] L. Devroye, L. Györfi, and G. Lugosi, *A Probabalistic Theory of Pattern Recognition*, ser. Applications of Mathematics: Stochastic Modeling and Applied Probability. New York: Springer-Verlag, 1996, no. 31.

[3] N. Gilbert, M. den Besten, A. Bontovics, B. G. W. Craenen, F. Divina, A. E. Eiben, R. Griffioen, G. Hévízi, A. Lôrincz, B. Paechter, S. Schuster, M. C. Schut, C. Tzolov, P. Vogt, and L. Yang, "Emerging artificial societies through learning," *Journal of Artificial Societies and Social Simulation*, vol. 9, no. 2, 2006.

[4] B. Rohrer, J. Morrow, F. Rothganger, and P. Xavier, "Concepts from data," in *Proceedings of the Brain-Inspired Cognitive Architectures Symposium, AAAI Fall Symposium Series*, 2009.

[5] W. T. B. Uther and M. M. Veloso, *Adaptive Agents in Multi-Agent Systems: Adaptation and Multi-Agent Learning*, ser. Lecture Notes in Computer Science. Springer, 2003, vol. 2636, ch. T Tree: Tree-based state generalization with temporally abstract actions, pp. 266–296.

[6] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Belmont, California: Wadsworth International Group, 1984.

[7] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[8] L. Rokach and O. Maimon, *Data Mining with Decision Trees: Theory and Applications*, ser. Machine perception and artificial intelligence, H. Bunke and P. S. P. Wang, Eds. Singapore: World Scientific Publishing Co., 2008, vol. 69.

[9] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," California Institute of Technology, Tech. Rep. 7694, 2007, http://authors.library.caltech.edu/7694.

[10] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *Journal of Physiology*, vol. 160, pp. 106–154, 1962.

[11] A. W. Moore and C. G. Atkeson, "The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces," *Machine Learning*, vol. 21, pp. 199–233, 1995.

[12] D. Chapman and L. P. Kaelbling, "Input generalization in delayed reinforcement learning," in *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, 1991, pp. 726–731.

[13] A. K. McCallum, "Reinforcement learning with selective perception and hidden state," Ph.D. dissertation, University of Rochester, Computer Science Department, 1995.

[14] W. T. B. Uther and M. M. Veloso, "Tree based discretization for continuous state space reinforcement learning," in *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-98)*, Madison, WI, 1998.

[15] M. J. Kochenderfer, "Adaptive modelling and planning for learning intelligent behaviour," Ph.D. dissertation, University of Edinburgh, School of Informatics, 2006.

[16] L. D. Pyeatt and A. E. Howe, "Decision tree function approximation in reinforcement learning," in *Proceedings of the Third International Symposium on Adaptive Systems: Evolutionary Computationand Probabalistic Graphical Models*, 2001, pp. 70–77.

[17] C. J. C. H. Watkins and P. Dayan, "Technical note: Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, May 1992.

[18] D. George and J. Hawkins, "A hierarchical bayesian model of invariant pattern recognition in the visual cortex," in *Proceedings of the IEEE Intl Joint Conference on Neural Networks*, vol. 3, 31 July-4 Aug 2005, pp. 1812–1817.

[19] G. Carpenter and S. Grossberg, "A massive parallel architecture for a self-organizing neural pattern recognition machine," *Computer Vision, Graphics, and Image Processing*, vol. 37, no. 1, pp. 54–115, 1987.